Métropole – 2024 – sujet1 - Correction

Exercice 3 (6 points)

---

1. Associer les termes

- `Region` → **classe**
- `nom`, `tab_voisines`, `tab_couleurs_disponibles`, `couleur_attribuee` → **attributs**

👉 *Commentaire : Une classe définit un modèle. Les attributs sont des variables stockées dans chaque objet (instance) de la classe.*

---

2. Type du paramètre `nom_region`

Type : **str** (chaîne de caractères)

👉 *Commentaire : L'énoncé précise : `param nom_region (str)` et on l'associe au nom d'une région.*

---

3. Création d'une instance

```
ge = Region("Grand Est")
```

👉 *Commentaire : On instancie la classe en appelant son constructeur avec le nom de la région.*

---

4. Méthode `renvoie_premiere_couleur_disponible`

```
return self.tab_couleurs_disponibles[0]
```

👉 *Commentaire : On renvoie le premier élément du tableau, d'indice 0.*

---

5. Méthode `renvoie_nb_voisines`

```
return len(self.tab_voisines)
```

👉 *Commentaire : Le nombre de voisines correspond à la longueur de la liste* `tab_voisines`.

---

6. Méthode `est_coloriee`

```
if self.couleur_attribuee is None:
    return False
else:
    return True
```

👉 *Commentaire : Une région est coloriée si* `couleur_attribuee` *n'est plus égale à None.*

---

7. Méthode `retire_couleur`

```
if couleur in self.tab_couleurs_disponibles:
    self.tab_couleurs_disponibles.remove(couleur)
```

👉 *Commentaire : On vérifie la présence pour éviter une erreur avec remove().*

---

8. Méthode `est_voisine`

```
for reg in self.tab_voisines:
    if reg == region:
        return True
return False
```

👉 *Commentaire : On parcourt la liste des voisines et on compare chaque élément.*

---

9. Méthode `renvoie_tab_regions_non_coloriees`

```
tab = []
for reg in self.tab_regions:
    if not reg.est_coloriee():
        tab.append(reg)
return tab
```

👉 *Commentaire : On construit un nouveau tableau contenant uniquement les régions sans couleur.*

10.Méthode `renvoie_max`

a. Quand la méthode `renvoie_max` renvoie-t-elle None ?

Elle renvoie **None** lorsqu'il n'y a **aucune région non coloriée**.

👉 *Commentaire : Si la liste renvoyée par `renvoie_tab_regions_non_coloriees()` est vide, la boucle ne s'exécute pas et `region_max` reste à None.*

---

b. Particularités de la région renvoyée

Lorsque la méthode ne renvoie pas None :

La région n'est **pas encore coloriée**.
C'est celle qui possède le **plus grand nombre de voisines** parmi les régions non coloriées.

👉 *Commentaire : C'est une stratégie gloutonne : on colorie d'abord la région la plus contrainte.*

---

11. Méthode `colorie(self)`

```
def colorie(self):
    region = self.renvoie_max()
    while region is not None:
        # attribution de la couleur
        couleur = region.renvoie_premiere_couleur_disponible()
        region.couleur_attribuee = couleur
        # mise à jour des voisines
        for voisine in region.tab_voisines:
            if couleur in voisine.tab_couleurs_disponibles:
                voisine.retire_couleur(couleur)
        # région suivante
        region = self.renvoie_max()
```

👉 *Commentaire : L'algorithme applique une stratégie gloutonne : on choisit toujours la région la plus contrainte restante, on lui attribue la première couleur disponible, puis on retire cette couleur chez ses voisines.*

Remarque pédagogique

👉 *Cet exercice est une application directe du problème classique de coloration de graphe. La stratégie utilisée est une heuristique gloutonne vue en première (non optimale en général, mais efficace).*
